11. Trabajando con flujos y archivos

Programación - 1º DAM

Luis del Moral Martínez

versión 20.10

Contenidos del tema

1. Flujos
2. Archivos

1. Flujos

Concepto de flujo

- Un **flujo** (stream) consiste en una serie de datos que fluyen entre un **origen** y un **destino**.
- Las **conexiones** se establecen con operadores (<<, >>) y **funciones de E/S**
- Podemos pensar que un flujo es un canal de datos entre el **teclado** y el **programa**
- **En esencia, un flujo es una secuencia de caracteres**

1. Flujos

Tipos de flujos

- Existen diferentes tipos de flujos:
  1. **Flujo de texto**: secuencia de caracteres (incluye retornos de carro, saltos de línea)
  2. **Flujo binario**: secuencia de bytes entre diversos dispositivos

1. Flujos

Clases de flujo en E/S

- El archivo de cabecera **<iostream>** declara tres clases para flujos E/S estándar
- Las clases derivadas de la clase base **ios** se usan para procesar flujos de alto nivel
- La clase iostream se suele usar en operaciones corrientes de E/S (cin, cout…)

1. Flujos

Lectura de datos de tipo carácter

- Los caracteres se leen de uno en uno, según las siguientes reglas:
  1. Los **espacios en blanco** (espacio en blanco, tabulación, nueva línea…) se ignoran al usar >> en cin
  2. Los **valores numéricos** se pueden leer como caracteres (cada dígito es un carácter independiente)
- Cuando se leen cadenas, se producen anomalías con los espacios
  - Si una cadena tiene un espacio en blanco o palabras separadas, cout se detiene
  - La cadena es **truncada** a partir del primer espacio en blanco
  - El operador **>>** hace que el objeto cin termine la operación de lectura con el espacio en blanco
- El flujo **cin** tiene varias funciones miembro para procesar entrada de cadenas y caracteres

1. Flujos

Funciones get() y getline()

- La función **get()** está definida en la clase **istream**. Tiene dos usos:
  1. **cin.get()**: se utiliza para capturar un único carácter **c = cin.get()**
  2. **cin.get(cadena, n)**: se utiliza para capturar una cadena de **n -1** caracteres (hay que contar con **\0**)
- La función **getline()** permite leer cadenas completas incluyendo espacios en blanco:
  - **cin.getline(cadena, longitud+2, terminador)**:
    - Se tiene que tener en cuenta una **longitud + 2** debido **\n** y **\0**
    - El terminador por defecto es **\n** (si no se pone se asume **\n**)
- Abre los ficheros de ejemplo **11_01_flujos.cpp** y **11_02_flujos.cpp**

1. Flujos

Problemas de la función getline()

- La función **getline()** funciona bien si leemos cadenas de forma consecutiva
- Sin embargo, presenta problemas si la usamos después de leer un **número entero** con **cin**
- Existen varias formas de solucionarlo:
  1. Especificar un carácter de terminación o separación diferente a **\n**
  2. Limpiar la memoria intermedia (**buffer**) del teclado leyendo el carácter sobrante (**CRLF**)
  3. Usar una sentencia de lectura diferente (funciones de cadena **get()** y **fgets()**, definidas en **stdio.h**)
- Abre los ficheros de ejemplo **11_03_flujos_3.cpp** y **11_04_flujos_4.cpp**

2. Archivos

Archivos

- C++ utiliza flujos para gestionar **flujos de datos**, incluyendo los flujos de **entrada** y **salida**
- Un archivo es una **secuencia de bits** que se almacena en un dispositivo externo
- Los bits se interpretan según el **protocolo** de un sistema software (ASCII…)
- En C++, un archivo es un flujo externo (bytes almacenados en disco).
- Los archivos pueden abrirse para **salida** (almacenar datos) para **entrada** (lectura de datos)

2. Archivos

Apertura de archivos

- Para que el programa pueda leer o escribir en el disco, antes hay que abrir el archivo
- Existen distintas formas de abrir el fichero:
  - **Lectura**: se utiliza un objeto de la clase **ifstream**
  - **Escritura**: se utiliza un objeto de la clase **ofstream**
- Al igual que sucede con cin y cout, se deben abrir flujos diferentes para la escritura y lectura
- Se recomienda comprobar que el fichero se ha abierto con éxito antes de trabajar con él
- Abre los ficheros de ejemplo **11_05_archivos.cpp** y **11_06_archivos_2.cpp**

2. Archivos

E/S en archivos (1)

- Se puede hacer **E/S** directamente a archivos
- Se utilizará el fichero de cabecera **<fstream>**
- Podemos crear distintos tipos de flujo:
  - **Flujo de entrada**: **ifstream** entrada;
  - **Flujo de salida**: **ofstream** salida;
  - **Flujo de entrada y salida**: **fstream** entradaSalida;

2. Archivos

E/S en archivos (2)

- Una vez creado el flujo, se usará la función **open()** para asociarlo a un archivo

```
archivo.open(nombre, ios::in);
archivo.open(nombre, ios::out);
archivo.open(nombre, ios::in | ios::out);
archivo.open(nombre, ios::out | ios:binary);
archivo.open(nombre, ios::in | ios:binary);
```

Argumento	Modo
ios::in	Modo entrada
ios::add	Modo añadir
ios::out	Modo salida
ios::ate	Abrir y buscar fin del archivo
ios::nocreate	Genera un error si el archivo no existe
ios::trunc	Trunca el archivo a 0 si ya existe
ios::noreplace	Genera un error si ya existe el archivo
ios::binary	El archivo se abre en modo binario

2. Archivos

E/S en archivos (3)

- La función **close()** cierra el archivo abierto (¡siempre se deben cerrar los archivos abiertos!)
- Otras funciones miembro de los flujos **fstream**:
  - **good()**: devuelve un valor distinto de 0 si no existe ningún error en una operación de flujo
  - **fail()**: devuelve un valor distinto de cero si el flujo ha alcanzado el final del archivo
  - **eof()**: el operador ! determina el estado de error para determinar el final del archivo **!archivo.eof()**

2. Archivos

E/S en archivos (4)

- La **E/S de texto** puede realizarse con los operadores **<<** y **>>** y el flujo
- Sin embargo, en la **E/S binaria** se emplean las funciones **get()** y **put()**
  - La función **get()** lee el flujo de entrada byte a byte (carácter)
  - La función **put()** escribe un carácter en el flujo de salida
- Abre los siguientes ficheros de ejemplo:
  - **11_07_archivos_texto.cpp**
  - **11_08_archivos_binario.cpp**
  - **11_08_archivos_binario_2.cpp**

Créditos de las imágenes y figuras

Cliparts e iconos

- **Obtenidos mediante la herramienta web IconFinder** (según sus disposiciones):
  - Diapositiva 1
  - Según la plataforma IconFinder, dicho material puede usarse libremente (free comercial use)
  - A fecha de edición de este material, todos los cliparts son free for comercial use (sin restricciones)

Resto de diagramas y gráficas

- Se han desarrollado en PowerPoint y se han incrustado en esta presentación
- Todos estos materiales se han desarrollado por el autor
  - Si se ha empleado algún icono externo, este se rige según lo expresado anteriormente